MAILLE AUTHENTICATION

A Novel Protocol for Distributed Authentication

Andrew Fritz and Jehan-François Pâris[1]
University of Houston, Department of Computer Science, Houston, Texas 77204-3010

Abstract: We present a decentralized solution to the distributed authentication problem. Unlike current schemes, Maille does not rely on a set of dedicated servers. Each participating node has a set of trusted peers that act as replicated repositories for its public key. Whenever a node A wants to contact another node B, it sends messages to its peers, which forward them to their peers, and so on until they reach B's peers. These peers then reply with B's public key and return them back through the paths the requests originally took. To guarantee the independence of replies, each node along each path forwards one reply only. Because of this, each Byzantine failure can introduce at most one false response. If the same key value is asserted by a qualified quorum of replies, A accepts it as the true key of B. Otherwise authentication fails.

Keywords: authentication, distributed security, peer-to-peer, Byzantine failure, denial of service, certificate authority

[1] Supported in part by the National Science Foundation under grant CCR-9988390

1. INTRODUCTION

> Maille*: ... mesh, network, a coat of mail, ... A flexible fabric made of metal rings interlinked. It was used especially for defensive armor.*
> -W*ebster's 1913 Dictionary*

In distributed environments, centralized services can become bottlenecks, and are often targets for attack, both electronic and political. As a result, any service that resides at some small fixed number of locations has the potential to limit the ultimate performance, availability and security of the system in question. Any critical portion of the distributed system controlled by one organization can effectively give control of the entire system to that organization.

Existing authentication systems require centralized management by a single organization or well-defined collaboration among organizations. This leads to a weakness. The Verisign problem—where Verisign changed the way many root DNS servers worked without the permission of the standards organizations or community—is a real-world example. Placing control of a central service required by many organizations into the hands of a single organization places the controlling organization in a monopoly position where they can use their management right to control the larger group. For political reasons, this type of structure may not be acceptable in many collaborations. In addition, it is inherently weak against legal and political attack. No known centralized authentication system can prevent this problem.

To handle normal failures, such as network outages, most systems replication authentication credential to a small fixed set of authentication authorities, or accept the limited scalability and uptime of a central authentication service. If all authentication server replicas happen to be on the same link, or even segment of the larger network, then a single failure may take them completely offline.

Furthermore, centralized authentication services provide a clear target for denial of service (DOS) attacks. By preventing authentication, an attacker can take an otherwise robust distributed system offline. If no one can be authenticated because all or most of the authentication authorities are unavailable, then the entire system is effectively taken offline.

Several researchers (e.g. Reiter and Stubblebine, 1997; Kahn, 1998) have noted that independent corroboration of credential is an effective way to remove this single point of failure. To our knowledge, no existing authentication system actually makes use of multiple independent sources of key information beyond a set of defined authentication servers. This paper presents the Maille authentication protocol, a new authentication protocol that eliminates authentication as a single point of failure. All nodes participate as equals to provide multiple verifiably independent sources of public keys. Each node has established trust relationships with some peers. Any node in the network may use its trusted peer relationships to obtain the

public key of any other node in the network. When one node in the network wishes to authenticate another node, it attempts to acquire that nodes public key using its peer relationships. The protocol is structured so the requester can verify the independence of the sources of keys. It then uses a conventional nonce challenge against the node to establish one-way trust. If mutual trust is required, each node uses the protocol to establish the identity of the other, and a direct secure session can be established.

As a result, the Maille protocol provides a means to acquire the public key of an entity in a trusted manner without a trusted central authority. Since all nodes equally share authentication functions, there are no favored targets of attack in a Maille network. Maille acts as a completely distributed certificate authority.

The remainder of this paper is laid out as follows. Section 2 covers previous work. Section 3 defines the Maille-authentication protocol. Section 4 provides an analysis of Maille's vulnerability to attacks. Section 5 presents future work. Finally, section 6 has our conclusions.

2. PREVIOUS WORK

Kerberos (Steiner et al., 1988) is a centralized authentication system, designed to allow single-sign-on from trusted workstations. Kerberos based systems rely on a single or a small set of authentication servers. The Kerberos system uses a ticket scheme, which allows clients to authenticate against the Kerberos servers only once. Thereafter, for the lifetime of the ticket, no further authentication is required and services and other individuals can trust the ticket holder without having to know their key.

Kerberos does have several weaknesses. First, it is highly centralized, requiring one master server where all updates occur. Replication of the security information to other server will offload all authentication work, but cannot reduce the total amount of work the master server must do to update security information and to broadcast changes. Further, because Kerberos relies on a single master server for all changes, that server becomes a single point of failure from a hardware, software, security and political standpoint.

The KryptoKnight family of protocols (Bird et al., 1995) is designed for embedded devices and is optimized for speed and efficiency. It relies on a single, possibly replicated, authority to provide trusted keys and act as an intermediary during authentication for all clients. The main focus is on providing several protocols that allow the exchange of keys, challenges and responses to flow as efficiently as possible by allowing the use of information each of the parties may already have. The KryptoKnight protocol family does not address issues of scalability or how credentials are revoked. A Byzantine failure in an authority is catastrophic for all parties using that authority.

Public key infrastructure (PKI) (Adams and Lloyd, 1997) has become very popular for Internet commerce. It is also widely used in grid computing

as the basis for the Globus Security Infrastructure (GSI) (Foster et al. 1998). PKI relies on a hierarchy of certificate authorities (CA) for scalability. At the top is the root CA, which signs certificates for servers in the second level and so on, until the lowest-level CAs are used to establish the identity of outside entities such as web servers. Revocations are handled through certificate expiration dates and revocation lists.

Replication of CA ensures that most authentications will not be affected by a single failure. However, the higher up the hierarchy an authentication is required to go, the more likely a single failure is to prevent successful authentication. Caching prevents most interactions from requiring the root CA and other high level CA servers. Nevertheless, a Byzantine failure at the root level will lead to a complete loss of security. Failures at lower levels will result in security breach for only part of the system. Politically, the root CA is a single point of failure.

PGP (Zimmermann, 1995) is a system designed to let many individuals authenticate each other without a central authority. It provides a method of creating and distributing keys among small clique of users and for deciding to trust a key acquired from a third party. How much trust can be placed in a public key is directly related to how many intermediaries it went through. A single failure in any intermediary can result in an incorrect key being used. Because of its decentralized nature, there is no global single point of failure since unless key repositories are used. PGP does not provide any way for two unrelated parties to authenticate each other than through trusted key repositories, which will then be single points of failure.

PathServer (Capkun et al., 2002) is an extension to PGP that fills the role of the trusted key repository suggested by the PGP users guide. The server models the network topology and tries to find mutually independent sources for the key, ensuring that agreement can be reach about their validity. In this way the normal PGP authentication system is strengthened against single failures in key sources and keys can be used that otherwise would not be acceptable.

However, PathServer itself represents a single point failure, as the requesting user still must trust a single entity for the key, namely the PathServer. The user is free to verify the key, but this results in the user's computer functioning as a PathServer. Further, the method used for finding independent keys is graph analysis. This technique requires much global knowledge and an approximation to escape the NP-HARD nature of the problem. PathServer does not assure that keys are actually independent, just likely to be that way.

Coca (Zhou et al., 2002) is a distributed certificate authority that makes use of threshold public key cryptography. Available servers transmit their portion of the certified key to a delegated server selected by the client. If the delegated server can collect enough parts, a complete key can be assembled. Coca is specifically designed to prevent DOS attacks. Further, if t is the minimum number of components required to assemble a public key and n the total number of servers, t+1 Byzantine failures must occur before the

public key of a service is compromised and (n–t)+1 failures must occur to deny authentication.

Coca relies on a fixed set of cooperating servers that all have the public keys of all other servers. As the number of replicated servers increases to meet demand, the risk of at least one server being compromised increases. If the threshold used in the threshold cryptography is not increased to match the server pool size, then it become easier to compromise the keys Coca holds. Increasing the threshold for cryptography increases network and processor overhead as more servers and messages are required to collect enough key parts to perform authentication.

3. THE MAILLE PROTOCOL

We want to design a protocol that allows participating nodes to authenticate each other securely without any centralized point of failure, including management. Individual nodes must not need global knowledge of the network. The system should not be dramatically affected by multiple Byzantine failures and should experience graceful degradation when faced with DOS attacks or the loss of many participating nodes.

Maille has no centralized point of failure because it is a decentralized key distribution system made up of individual, equal and autonomous nodes. Each node in the network has a set of peer nodes. Each node's peers are explicitly trusted. Like in any large distributed authentication system, this trust is transitive. Each node trusts its peers, and all trust in other nodes is derived from this basic trust. A Maille node uses its trusted peer relationships to find a set of independent sources for key information. If enough completely independent sources agree on the key, then the acquired key can be used to challenge the other party to prove its identity via a nonce challenge.

Maille ensures that individual responses are independent by tracking the path (called a *response chain*) by which each response reached the requester. A response chain includes all nodes that relayed the message from the node holding a copy of the requested public key back to the requester. The protocol guarantees that non-independence chains will always be detected.

Maille also guarantees that each properly functioning node filters duplicates and extra responses. Thus, if n independent chains are found, the requester can be sure that n nodes independently returned chains with the same key. A single Byzantine failure can never introduce more than one false chain and one false key. In addition Maille requires a qualified quorum of independent agreeing chains. Hence many such failures in separate, but very specific parts of the network must occur to provide a false key.

3.1 Assumptions

1. Network transport is not secure unless Maille uses a secure channel it creates using keys it trusts.
2. Attackers know the network structure and can attack any machine in the network with equal effort.
3. Attackers know the protocol and can use that knowledge to find the most critical node(s) to any transaction.
4. Any node may fail or be compromised.
5. No single source of information can ever be trusted.
6. The underlying cryptography functions are strong. They cannot be broken easily without some other failure that provides clues.

3.2 Notations

In the following discussion, the notations listed below are used.

- A, B, C represent specific nodes in the network.
- X and Y represent any arbitrary node in the network.
- *f* (...) is a message, containing, amongst other, the parameters specified between the parentheses.
- *A* (in italics) represent the public key of node A.
- *Ø* is the special null key. It is treated exactly as any other key except where noted.

For example, A may send to B the message *kr*(A,C) to request the public key *C* for node C.

3.3 Node Structures

Each node in the network maintains the following:

- *kr_cache*: a cache of recent key requests storing the requester and a return seen flag (initially false).
- *trust_list*: a list containing the public keys of the peers of the node.
- *black_list*: a list of nodes that are blacklisted.

Note that *trust_list* and *black_list* must mutually disjoint.

3.4 Messages

- *kr*(A,C) denotes a key request that contains the ultimate source (A), the target (C). Other parameters include the remaining hop count and the immediate source (the peer that forwarded the message to the node).
- *trust*(*kr*(A,C),*C*) denotes a statement of trust by the sender to the receiver that *C* is the key requested by *kr*(A,C).
- *rkr*(...) denotes a return key request for the public key of C in the form: *trust*(*kr*(A,C),*C*) | *rkr*(B,*rkr*(...)) | *rkr*(B,*trust*(*kr*(A,C),*C*)), where B is the peer who sent the *rkr*(...).

3.5 Peer Relationships

Within the peer-to-peer network, each node has a set of trusted peers. Peer relationships are established offline from the point of view of the Maille protocol. However, there are several required features of peer relationships that the rest of the system relies on.

If two nodes A and B are peer nodes, then A has a trusted public key of B, and vice versa. These keys are only used to establish a bidirectional secure channel between A and B. For added security, each node has one key per peer so that the public key A uses to establish a secure channel with peer B is different than the public key A would use to establish a secure channel with peer C. The secure channel from A to B is called secure channel A-B.

How A and B exchange public keys initially (i.e. become peers), or how A and B chose each other as peers isn't the focus of this paper.

Upon startup, peers A and B mutually authenticate each other using the appropriate key pair and establish a secure channel for use later. This secure two-way channel is used to pass all messages from A to B and from B to A for the duration of the protocol. Thus, A can trust that any message arriving via the secure B-A channel came unaltered from B and only B and vice versa.

Each node A also maintains a key pair for third party authentications. The public key *A* is passed to the peers along with a version number. When a key request message arrives at a peer of A, it returns that key *A* to the node that requested it. An optional parameter *l* specifies the lifetime of the third party key pair. If additional protection against a combination of Byzantine failures and clear text cryptographic attacks is required, we may choose to require A to change its key pair from time to time.

3.6 Obtaining Keys

The Maille protocol is made up of three individual protocols, each executed in response to particular types of messages. The key request initiator protocol is used by nodes to find keys. The key request forwarding protocol controls the propagation of *kr*(…) messages in Maille and the return key request protocol governs the propagation of *rkr*(…) messages.

3.6.1 Key Request Initiator Protocol

When node A, which is part of the network, wishes to obtain a trusted public key for node C, which is also part of the network, it does the following:

1. Node A initiates a key search by sending to its peers *kr*(A,C).
2. It waits for *rkr*(...) responses from them.

3. When a response arrives, node A checks that the sending peer has not already returned a response. If it has, it drops this response and return to step 2.
4. Node A creates a new *rkr*(…): *rkr*(Z, *rkr*(…)) where Z is the peer that returned the original *rkr*(…).
5. Node A adds the new *rkr*(...) to the set of chains received and returns to step 2.

3.6.2 Key Request Forwarding Protocol

Any node X receiving a key request *kr*(A,C) from some peer node Y carries out the following procedure:

1. Node X verifies that the key request *kr*(A,C) is not in its *kr_cache*. If it is, it drops this instance of *kr*(A,C) and stops.
2. Node X checks whether it has in its *trust_list* a trusted public key *C* of node C in *trust_list* whose lease has not expired. If it has such key, it returns an *rkr*(*C*) to node Y in the form *trust*(*kr*(A,C),*C*) and stops.
3. Node X checks if there is an entry in *black_list* for C. If there is, it returns *rkr*(Ø) to Y in the form *trust*(*kr*(A,C),Ø)) and stops.
4. Node X subtracts one from the remaining hop count of *kr*(A,C). If the hop count is now 0, it drops *kr*(A,C) and stops.
5. Node X adds *kr*(A,C) to the node's *kr_cache*.
6. Node X sends *kr*(A,C) to all node's peers, except node Y.

3.6.3 Return Key Request Forwarding Protocol

Any node X receiving an *rkr*(...) message from a node Z does the following:

1. Node X checks whether *kr_cache* contain an entry that matches *kr*(A,C) of this *rkr*(...) message. If it does not, node X drops this *rkr*(…) message and stops.
2. If the *kr_cache* entry has the *seen flag* set, node X drops this *rkr*(…) message and stops. This ensures that node X will only return only one *rkr*(…) per *kr*(…). Otherwise two dependent chains would be returned from node X.
3. Node X marks the *kr_cache* entry for *kr*(A,C) of this *rkr*(...) message as *seen*.
4. Node X returns a new *rkr*(…) message to the immediate source of the *kr*(…) in *kr_cache*, node Y, in the form *rkr*(Z,*rkr*(...)) to verify that it received an *rkr*(…) message from Z over the secure Z-X channel.

3.6.4 Protocol Outcome

Carrying out this protocol will result in A receiving a set of *rkr*(…) messages (called chains).The *kr*(A,C) might cause A to receive a chain like:

rkr(A,*rkr*(B,*rkr*(X,*trust*(*kr*(A,C),*C*))))
which is more easily under stood in the form A>B>X>Y with key *C*

This means that A received an *rkr*(…) from B, who received an *rkr*(…) from X who received the key *C* from Y. Note that no node adds itself to the chain. It is added instead by the node to which it returns the *rkr*(…).

3.7 Picking a Winning Key

Whenever distinct conflicting keys are returned, we must decide which key to accept as the public key of C. This procedure is carried out even if only one distinct key is received. The score it produces is required in later steps.

Chains are separated into groups based on the public key they contain C_1, C_2, and so forth. Each unique key will receive a score that is the total of all the scores of all chains that contain that key. The score of each individual; chain is computed by raising a link trust factor $0 < w \leq 1$ to the power of the length of the chain $|rkr(...)|$:

$$w^{|rkr(\ldots)|}$$

The key with the highest total score is the only one considered.

3.8 Independence Analysis and Penalties

The wining key is not necessarily the correct key because rogue nodes could have flooded their peers with the incorrect key. Therefore an independence analysis is performed on all chains containing the winning key. The analysis must determine the number of collisions that occur. A collision is defined as the number of times any two chains contain the same node, or 1 minus the number of times each distinct node appears in any chain.

The number of collisions is used as a penalty. If all chains are independent, then the number of collisions is 0. Otherwise, the number of collisions is subtracted from the total score the key received in the previous step.

3.9 Determining if the Winning Key Should be Trusted

If the adjusted score of the winning key surpasses some threshold of trust *t*, that key is used. If that key is the null key (Ø) then C is added to the *black_lis*t of node A and authentication fails. Otherwise the key request fails and no authentication may occur. Node A may choose to retry the key request with a longer hop count.

3.10 Using Keys

Once node A has obtained a key *B* for node B, it will use this key to authenticate B by sending a one-way nonce challenge. If B wishes to authenticate A as well, it will need to acquire *A* and extend the challenge to a two-way challenge.

3.11 Tunable Parameters

Maille includes several tunable parameters. All the parameters of Maille are related in intuitive ways to the level of security those using Maille want.

3.11.1 Chain Scoring

The link trust factor *w* determines how much trust is placed on each hop a response takes. A value of *w* near 1 will result in longer chains being nearly as trusted as shorter chains. A value of 1 will result in all chains being trusted equally. A value below and near 1 is suggested.

The trust threshold *t* is used to determine if a key has amassed a qualified quorum of support. The value of *t* will need to be less than the minimum peer count *m*. If *w* is 1, that is, all chains are treated equally regardless of length, then one key with all *m* peers supporting it will receive a score of *m*. If *t* is above *m* the no key can satisfy the quorum.

Further, if *t* is too close to *m*, many keys may not be able to amass enough support to be trusted. Higher values of *t* will make incorrect authentication of rogue nodes less likely, but will also allow fewer rogue nodes to launch a successful DOS attack.

One good compromise would be to set *t* relative to the actual peer count a node has. It could be set to 2/3 of the total peer count, thus requiring 2/3 of all peers to return chains that support one key for the key to be trusted.

3.11.2 Peer Relationships

The minimum peer count *m* determines how strong the system is in general, assuming all nodes have not many more peers than *m*. If *m* is only 3 (and all nodes have exactly 3 peers), *w* is 1, and *t* is 2 then a minimum of only 2 Byzantine failures is required to cause a false key to win the competition. If *m* is raised to 30 and *t* to 20, 20 such failures are required for the false key to win, and 11 are required to deny authentication that should otherwise occur.

4. ANALYSIS

The Maille authentication protocol removes the centralized authority that normally acts to dispense trusted public keys. Maille's design also changes the system dynamics so that authentication experiences graceful failure, both in terms of availability and security. As nodes fail, or are compromised, trust is not completely lost. Because all nodes are operated separately by the participating organizations and participate equally and there is no hierarchy of nodes within the network and no single control point, no one organization is placed in a monopoly position.

4.1 Byzantine Failures and Impersonation

Consider that a node A wishing to authenticate some other node claiming to be C. To do this, it must acquire C's public key *C*. Should a node D try to impersonate C, it must either acquire C's private key, which only C has, or cause A to receive and trust its public key *D* instead of *C*. To prevent the first problem, C must protect its private key. We consider only the second case.

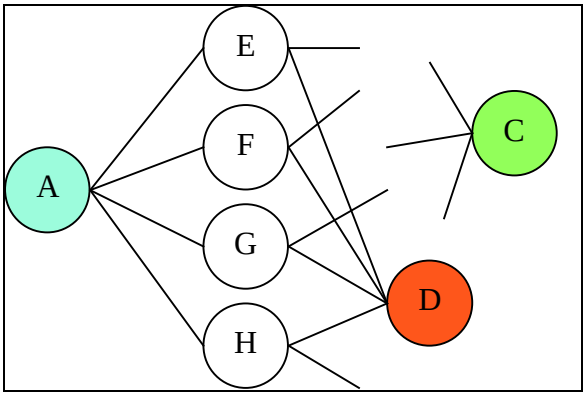

Figure 1. A is the requestor; C is the real target; D is the node that wishes to impersonate C; E, F, G and H are A's peers.

D may choose to operate alone or with others to provide incorrect keys. If it operates alone, it must provide enough responses to A that favor *D* instead of *C*. As in figure 1, the worst case occurs when D is peered with all of A's peers E, F, G and H. It could then fabricate an *rkr*(…) message that provided *D* and quickly send it to E, F, G and H. Because the fake *rkr*(…)s from D were received by A's peers before all other replies, they will be the only ones to reach A. If *w* is 1, *D* will receive a score of 4. However, during the independence analysis, 3 collisions would be found because E, F, G and H would all add D to the *rkr*(…) they forward. The key *D* will only receive a

score of 1. If the trust threshold *t* is set appropriately, this will not be enough for A to trust *D*.

D may also choose to collaborate with other rogue nodes to trick A into accepting *D*.

Looking back at figure 1, we see that A may trust the wrong key if enough of its peers act as collaborating rogues. To trick A to accept the wrong public key for C, the number of peers f_a returning the same false key must satisfy the inequality

$$f_a \geq t/w$$

If *w* is 1, at least *t* of A's peers must be collaborating rogues for A to use the wrong key. For a small value of *m*, as in the example above, only a small number of rogues is required. The easiest counter measure is to select a value of $w<1$, which will increase minimum number of rouges required to fool A. Unfortunately, it will also reduce the likelihood a valid key will achieve a qualified quorum. In larger networks with larger values of *m*, *t* can be higher and many more collaborating rogues will be needed. If the rogues are not directly peered with A, more collaborating rogues may be required to fool A.

4.2 DOS Attacks

Rogue nodes may wish to exploit the Maille protocol to deny successful authentication. System without established peer relationships (i.e. computers that are not part of a Maille network) have no chance to exploit the protocol since all messages within the protocol flow over secure channels between peers. Outside nodes cannot insert messages.

Rogue nodes can produce messages that try to exploit flaws in the protocol. Consideration should be given to various possibilities. To prevent A from obtaining a trusted public key for C, enough messages must be suppressed to prevent *C* from scoring higher than *t*. This would require a partial partitioning of the network such that there were less than *t* independent paths from A to C (assuming $w = 1$). However, as the number of paths decreases towards *t* it will become less likely due to the random routing used in Maille that A will receive enough independent replies to trust any key.

Rogue nodes could also exploit the independence analysis by providing a correct key with a fabricated chain. The artificial chain could be created to contain many of the nodes that might take part in the real chains. This would artificially penalize otherwise valid responses.

5. FUTURE WORK

We plan to extend the Maille protocol in several fashions. First, we plan to investigate techniques to automatically detect and blacklist rogue nodes. Second we want to develop protocols for adding and readmitting nodes to the network. Third we are designing an authorization protocol to work along side the Maille authentication protocol.

We are also currently building a simulation model that will allow us to explore the scalability of the system, routing, failure detection, network structure and restructuring.

6. CONCLUSIONS

The Maille authentication system is a completely distributed authentication system based on established asymmetric cryptography and nonce challenges. In some respects, it is similar to the certificate authorities of PKI because it acts as certificate authority to facilitate authentication. However, Maille authentication is designed to work in an environment where the assumptions made by most systems are unacceptable for political or security reasons. Maille is hardened against denial of service attacks, a reality of modern Internet computing. Because of its totally decentralized nature, Maille resists Byzantine failures, natural failures and organizational subversion. Its performance and security degrades gracefully in the face of failures or compromises instead of experiencing total collapse in the face of a small number of failures.

A Maille network is a peer-to-peer network where all nodes participate equally. Any node may use its peer relationships to acquire the trusted key of another node in the network. To do so, the Maille protocol is designed to find independent sources for the key of the target, and to maintain traceability of all who took part in finding the key. This combination allows the requesting node to know with a high degree of certainty that the key is authentic and for the target. Because the task of providing keys is not trusted to any one entity or small group of entities, one organization cannot use control of authentication to subvert the larger system.

REFERENCES

Adams, C., and Lloyd, S., 1997, Profiles and Protocols for the Internet public-key infrastructure, *Proc. 6th IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 220–224.

Bird, R., Gopan, I., Herzberg, A., Janson, Ph., Kutten, S., Molva, R., and Yung, M., 1995, The KryptoKnight family of light-weight protocols for authentication and key distribution, *ACM/IEEE Transactions on Networking*, **3**(1):31–41.

Capkun, S., Buttyan, L., and Hubaux, J.-P., 2003, Small worlds in security systems: an analysis of the PGP certificate graph, *Proc. 2003 ACM Workshop on New Security Paradigms*, pp. 28–35.

Foster, I., Kesselman, C., Tsudik, G., and Tuecke, S., 1998, A security architecture for computation grids, *Proc. 5th ACM Conference on Computer and Communication Security*, pp. 83–92.

Kahn, C., 1998, Tolerating penetrations and insider attacks by requiring independent corroboration, *Proc. 1998 ACM Workshop on New Security Paradigms*, pp. 122–133.

Reiter, M. K., and Stubblebine, S. G., 1997, Path independence for authentication in large-scale systems, *Proc. 4thACM Conference on Computer and Communications Security*, pp. 57–66.

Steiner, J, G., Neuman, C., and Schiller, J. I., 1988, Kerberos: an authentication service for open network systems, *Proc. 1988 Winter Usenix Conference*, pp. 191–201.

Zhou, L., Schneider, F. B., Van Renesse, R., 2002, COCA: A secure distributed online certification authority, *ACM Transactions on Computer Systems*, **20**(4):329-368.

Zimmermann, P., 1995, *The Official PGP User's Guide.* MIT Press, Cambridge, MA.